Collections

ArrayList

Les méthodes

1) add(Object o): ajoute un élément à la fin.
list.add("Bonjour");

2) add(int indice, Object o): Insertion au milieu.

```
list.add(3, "e");
```

Elle insère la chaîne de caractères String dans la troisième position.

3) addAll(Collection c): Ajoute une collection.

```
ArrayList l1 = new ArrayList();
l1.add("mot");
l1.add(12);
l1.add(10.4f);
list.addAll(l1);
```

Cette méthode ajoute une autre liste l1 à la fin de la liste list.

4) addAll(int indice, Collection c): Ajoute une collection au milieu.

```
list.addAll(3, l1);
```

Insertion de la liste l1 dans la position 3. Les autres éléments qui ont un indice supèrieur à 3 seront décalés.

5) clear(): Supprimer tous les éléments.

6) contains(Object o): Appartenance d'un élément.

```
boolean b = list.contains(o)
```

La valeur de b vaut true si l'objet o appartient à la liste.

8) ensureCapacity(int capacite): augmente la capacité, elle assure qu'elle peut contenir un nombre d'éléments avec sa capacité minimal.

```
list.ensureCapacity(10);
```

Ceci va augmenter la capacité avec 10 éléments.

9) get(int indice): retourne l'objet à une position.

```
system.out.println(list.get(2));
```

Cette fonction affiche le deuxième objet stocké dans la liste.

10) indexOf(Object o): Retourne l'indice d'un élément.

```
int k = indexOf("o4");
```

La variable entier k va recevoir l'indice de la première occurrence de l'objet o4.
On a dit bien juste la première occurrence.

11) isEmpty(): Retourne true si la liste est vide.

```
boolean b = list.isEmpty();
```

Si la liste est vide b est true.

12) remove(Object o): supprime la première occurrence de l'objet o.

```
boolean b = list.remove("o4");
```

Retourne true si l'objet existe et a été supprimé avec succès.

13) removeAll(Collection<?> c): supprime les éléments qui sont dans la collection passé en argument.

```
AarrayList<String> lc = new ArrayList<String>();
lc .add("o1");
lc .add("o2");
lc .add("o3");
list.removeAll(lc);
```

Elle cherche les éléments et les supprime.

14) removeRange(int début, int fin): supprime les éléments qui se situent entre l'indice début et l'indice fin.

```
list.removeRange(3,8);
```

Cette procédure supprime les éléments entre 3 et 8.

15) retainsAll(Collection<?> c): garde seulement les éléments qui sont dans la collection c.

16) set(int indice, Object o): modifie l'objet dans un position spécifique.

```
list.set(3, "o5");
```

L'objet dans la position 3 a été remplacé par o5.

17) size(): retourne le nombre d'éléments.

18) subList(int début, int fin): retourne le fragment situé entre le début et la fin.

19) toArray(): retourne un tableau d'une dimension.

```
String[] t = list.toArray();
```

Le tableau t contient tous les objets de list. Cette méthode est utile lorsque on a une fonction qui n'accepte que les tableaux par exemple.

20) trimToSize(): réduit la capacité de stockage au niveau maximum.

Comment parcourir un ArrayList

On peut parcourir un ArrayList avec deux méthodes:

1) Boucle for

```
for(int i = 0; i < list.size(); i++)
   system.out.println(list.get(i));
ou par exemple si on connait le type:
for(Integer nombre : list)
   system.out.println(nombre);
```

2) Iterator + While

```
Iterator itr = list.iterator();
while(itr.hasNext())
     system.out.println(itr.next());
```

Exemple

```
import java.util.ArrayList;

public class Test {

 public static void main(String[] args) {

  //créer un arraylist avec une capacité initiale de 4
  ArrayList str = new ArrayList(4);

  //Ajout
  str.add("o1");
  str.add("o2");
  str.add("o3");
  str.add("o4");

  //Quelques méthodes qu'on a vu
  System.out.println("indice de "+"o2: "+str.indexOf("o2"));
  System.out.println("o3 existe ? "+str.contains("o3"));
```

```
  System.out.println("o2 supprimé avec succès: "+str.remove("o2"));
  System.out.println("taille: "+str.size());
  System.out.println("[1, 3] : "+str.subList(1, 3));

  //parcours
  for(String s : str)
   System.out.println(s);

  str.clear();
  System.out.println("liste est vide ? "+str.isEmpty());

 }
}
```

Méthodes de la classe LindekList

1) void add(Object o): ajoute un élément à la liste.

```
list.Ladd("bonjour");
```

2) void add(int indice, Object o) : ajoute un élément à une position définie.

```
list.add(3, "position3");
```

3) void addAll(Collection c): ajoute les éléments d'une autre collection de données comme ArrayList. Elle lève une exception **NullPointerException** si la collection est nul.

```
LinkedList linkedlist = new LinkedList();
ArrayList arraylist= new ArrayList();
arraylist.add("123");
arraylist.add("456");
linkedlist.addAll(arraylist);
```

4) void addAll(int indice, Collection c): ajoute les éléments d'une autre collection de données comme ArrayList en commençant d'une position

donnée. Elle lève une exception NullPointerException si la collection est nul et IndexOutOfBoundsException si vous avez dépassez la capacité de la liste.

```
linkedlist.addAll(3, arraylist);
```

5) void clear(): efface le contenu de la liste.

```
list.clear();
```

6) Object clone(): retourne une copie de la liste.

```
System.out.println("linkedList: "+list);
Object str= list.clone();
System.out.println("str: "+str);
```

Sortie:

```
linkedList: [object1, object2, object3]
str: [object1, object2, object3]
```

7) boolean contains(Object o): elle vérifie si l'objet est présent dans la liste. Si l'élément existe, elle retourne true sinon false.

```
boolean var = list.contains("String");
```

8) Object get(int indice): retourne l'élement à l'indice donné.

```
Object elt = llist.getLast();
```

9) int indexOf(Object o): retourne l'indice d'un objet donné.

```
int pos = llist.indexOf("o2");
```

10) int lastIndexOf(Object o): retourne l'indice de la dernière occurrence d'un objet donné.

```
int lastpos = llist.lastIndexOf("o6");
```

11) Object remove(int indice): supprime un objet à l'indice donné.

```
llist.remove(4);
```

12) Object remove(Object o): supprime un objet spécifique de la liste.

```
llist.remove("o6");
```

13) Object removeFirstOccurrence(Object o): supprime la première occurrence rencontrée.

```
llist.removeFirstOccurrence("bonjour");
```

14) Object removeLastOccurrence(Object o): supprime la dernière occurrence rencontrée.

```
llist.removeLastOccurrence("bonjour");
```

15) Object set(int indice, Object o): modifier la valeur d'un élément à un indice spécifique.

```
llist.set(llist.sise()-1, "bonsoir");
```

Mettre "bonsoir" dans la dernière position de la liste, on a mis "-1" pour ne pas dépasser la taille de la liste.

16) int size(); retourne la taille actuelle ou le nombre total des objets présents dans la liste.

```
llist.size();
```

Méthodes propres à LinkedList

1) void addFirst(Object o): insère un élément dans la première position.

```
list.addFirst("string");
```

2) void addLast(Object o): insère un élément dans la dernière position.

```
list.addLast("string");
```

3) Object getFirst(): retourne l'élément à la première position.

```
Object elt = llist.getFirst();
```

4) Object getLast(): retourne l'élément à la dernière position.

```
list.addFirst("string");
```

5) void removeFirst(): supprime l'élément de la première position.

```
list.removeFirst();
```

6) void removeLast(): supprime l'élément de la dernière position.

```
list.removeLast();
```

Exemple de LinkedList

Cette exemple illustre différents méthodes les plus populaires supportées par LinkedList:

```
package LinkedList;
import java.util.*;

public class LinkedListDemo {

   public static void main(String args[]) {
      // déclaration de linked list
      LinkedList ll = new LinkedList();
      // remplir les éléments dans linked list
      ll.add("C");
      ll.add("D");
      ll.add("T");
```

```
      ll.add("V");
      ll.addFirst("A");
      ll.addLast("Z");
      ll.add(1, "B");

      System.out.println("Contenu original: " + ll);

      // remove elements from the linked list
      ll.remove("F");
      ll.remove(2);
      ll.removeFirst();
      System.out.println("Après suppression: "
       + ll);

      // modification de la valeur de l'objet à l'indice 3
      String first = ll.getFirst();
      int index = ll.indexOf(first);
      ll.set(index, first + " Nouveau");
      System.out.println("Aprés modification: " + ll);
   }
}
```

Sortie:

```
Contenu original: [A, B, C, D, T, V, Z]
Après suppression: [B, D, T, V, Z]
Aprés modification: [B Nouveau, D, T, V, Z]
```